CORRECT

Lab 4

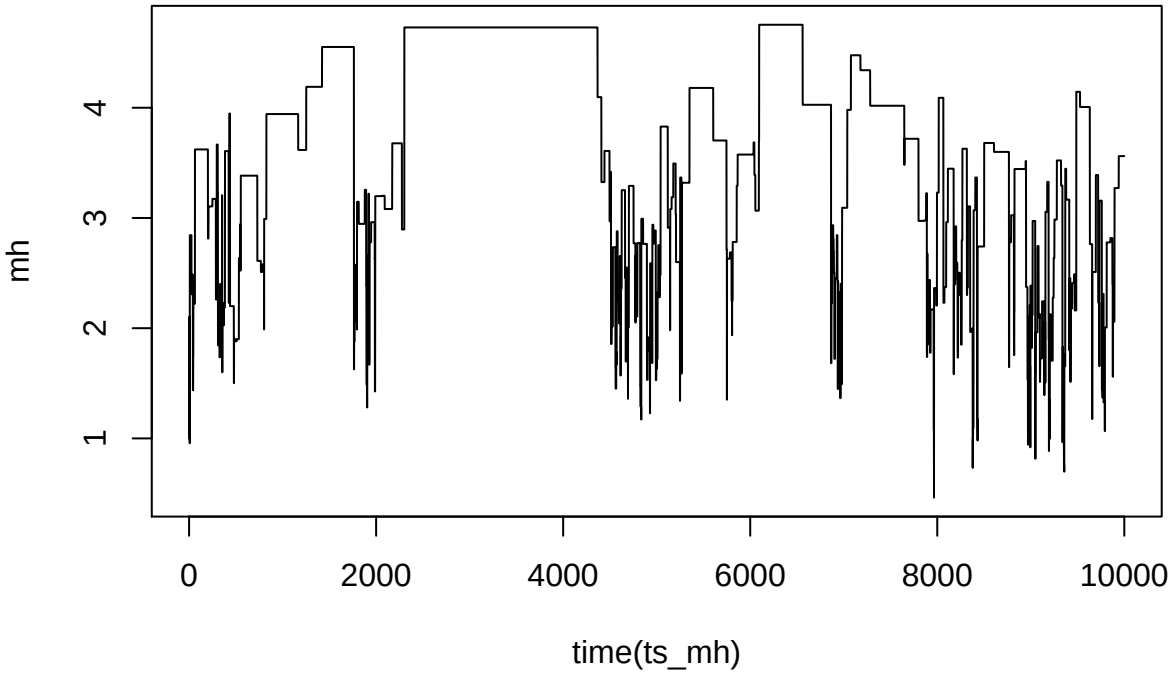
Niclas Lovsjö, Maxime Bonneau

1 mars 2016

Assignment 1

1.

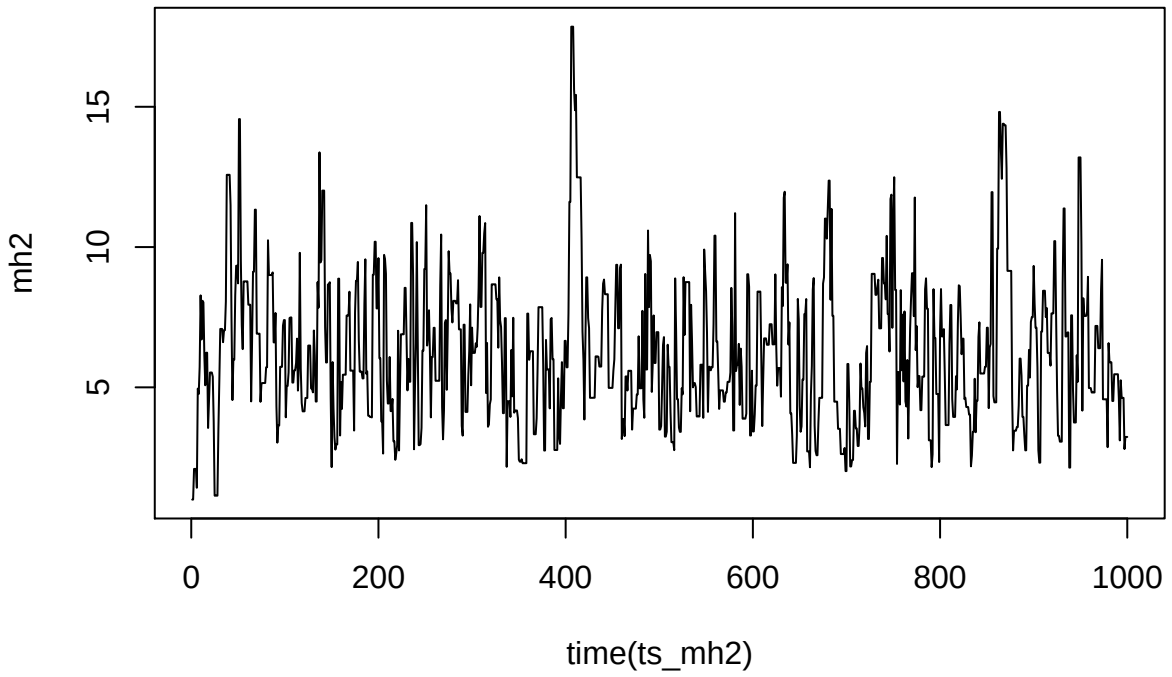
First we will generate a sample from the proposed distribution, thanks to the Metropolis-Hastings' algorithm. Here is the time serie of this method, with a starting point of 1:



We can notice some plateaus, but absolutely no convergence in this case. So there is no burn-in period.

2.

For this question, I repeated the same step, but with a chi-square distribution as proposal distribution instead of a log-normal distribution.



This time we can notice a convergence, which seems to be actually quite fast. The burn-in period (ony by looking at the graph) seems to be around the 20 first values.

3.

We can conclude by comparing these two examples that depending on the proposal distribution, the convergence to the probability density function is more or less fast. For the first case, there is absolutely no pattern after 10000 iterations, so we can guess that quite a lot of time is needed to fit well the proposal distribution. In opposition, the second proposal distribution achieves a really fast burn-in period.

4.

To analyze the convergence of the sequences starting by points from 1 to 10, we need to use Gelman-Rubin method. The output of the function in coda library is:

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.01
```

We can see that the value "Upper C.I." is 1, which means that the convergence is achieved.

5.

We would like to estimate $\int_0^\infty x.f(x)dx$ using samples from step 1 and step 2.

If $\theta = \int_0^\infty g(x)f(x)dx$, then $\hat{\theta} = \frac{\sum g(x_i)}{m}$.

Here g(x) = x. So $\hat{\theta} = \frac{\sum x_i}{m}$.

With values from step 1, $\hat{\theta} = 3.6771171$.

With values from step 2, $\hat{\theta} = 6.2909808$. (The values of the 20 first observations are not used, since it looks to be the burn-in period).

6.

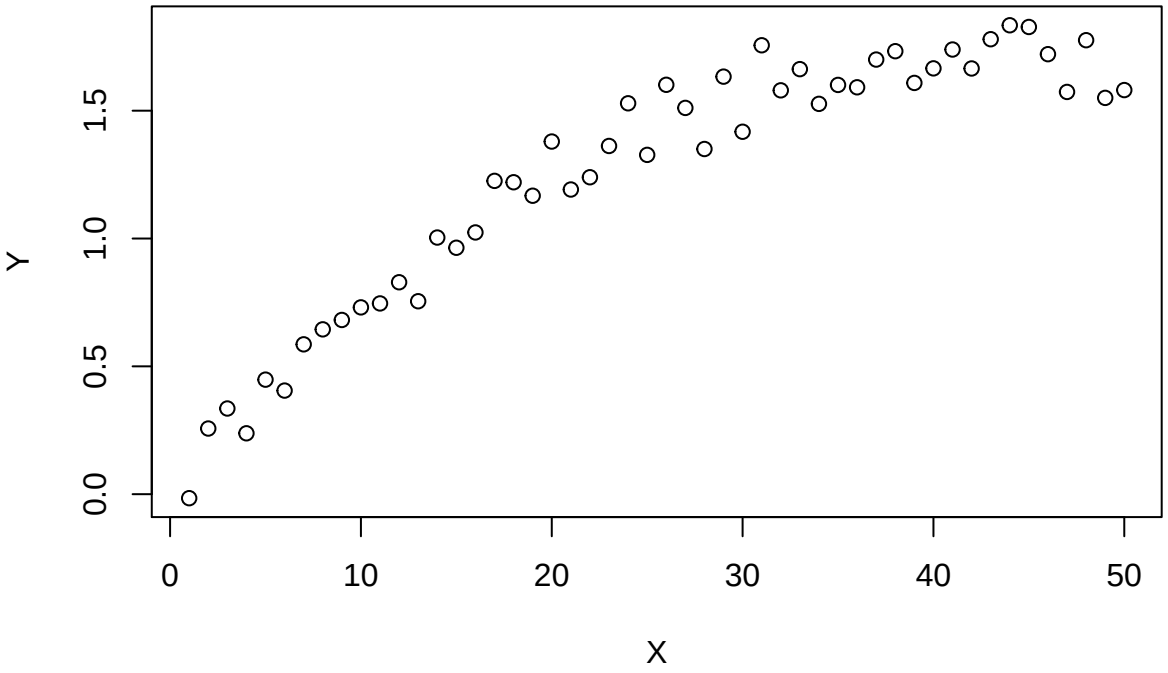
If the probability density function follows a gamma distribution, it is a $\gamma(6, 1)$.

So the real value of the integrale is 6, which is closer to the value found for the second proposal distribution. This is one more indication that this proposal distribution is better.

Assignment 2

2.1)

Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?



We assume that some linear model could be used here.

2.2)

Present the formulas for $P(Y|\mu)$ and $P(\mu)$:

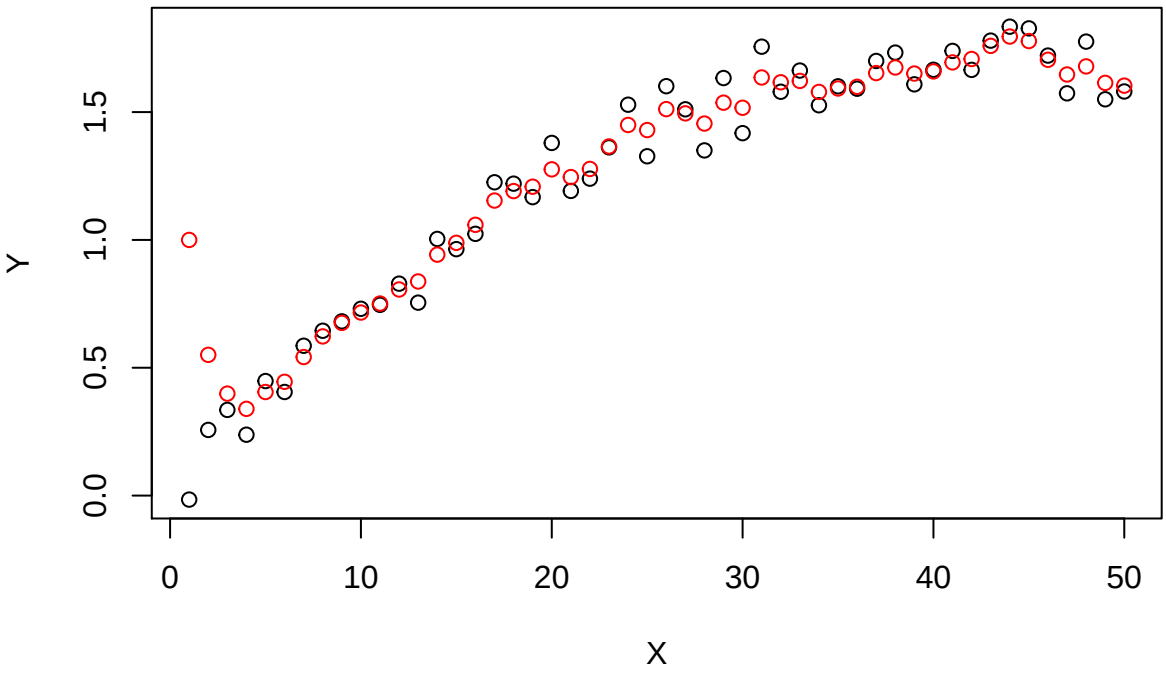
The formulas is written as, $p(\mu|Y) = p(\mu)p(Y|\mu) = \alpha\ exp(-\frac{\sum_{i=1}^{n}(Y_i-\mu_i)^2+\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2}{2\sigma^2})$

2.3)

Each distribution $\mu_i|\mu_{i-1}$ is proportional to $exp(-\frac{\mu_i^2+\frac{2}{3}\mu_i(Y_i+\mu_{i-1}+\mu_{i+1})}{\frac{2}{3}\sigma^2})$ and *completing the square* gives that we should generate normals having mean $\frac{1}{3}(\mu_{i+1} + Y_i + \mu_{i-1})$.

2.4)

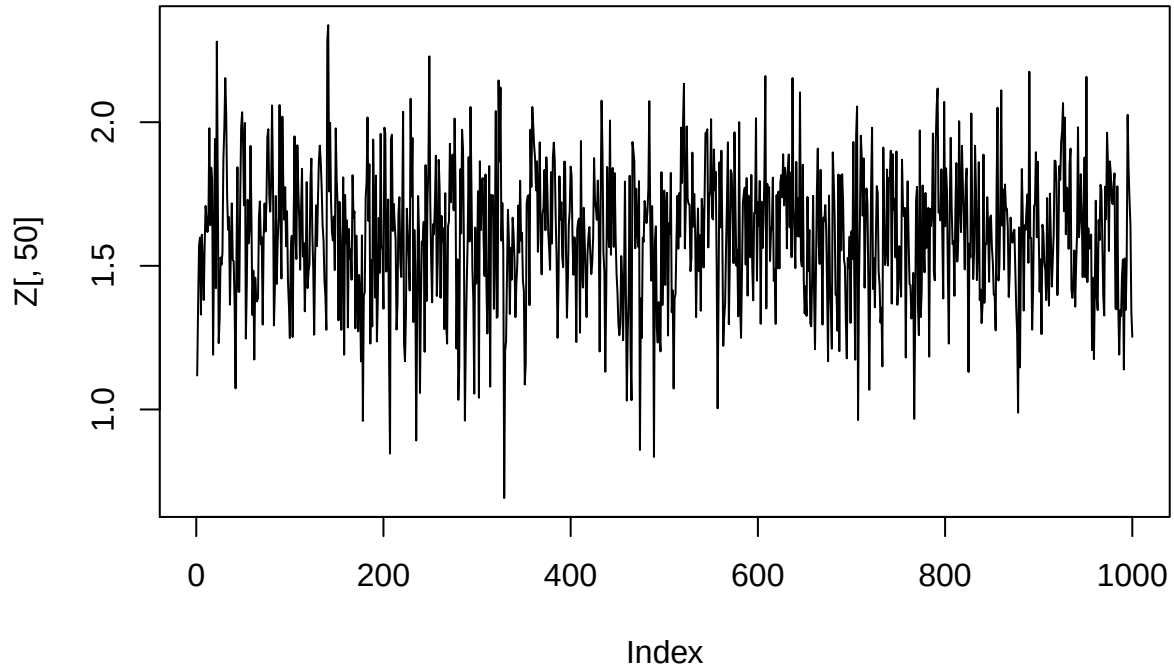
We use the results from 2.3 and implement a Gibbs sampler, then we use the Monte Carlo approach to get the expected value of μ. This yields the following plot of μ vs X and Y vs X.



We see that the red dots seem centered compared to the black dots, which means that we seem to have reduced the noise and that the generated μ has caught the underlying dependence.

2.5)

Finally we make a trace-plot of μ_{50}.



We see that the first couple of values seem lower than the rest, we expect we something like 25 iterations as burn-in period.

Code:

```
library(knitr)
opts_chunk$set(echo=FALSE)
library(TSA)

pi <- function(x){
  p <- x^5*exp(-x)
  return(p)
}

alpha_ln <- function(x, y){
  q_x_y <- dlnorm(x, mean = y)
  q_y_x <- dlnorm(y, mean = x)
  a <- min(1, pi(y)/pi(x)*q_x_y/q_y_x)
  return(a)
}

mh <- function(init = 1){
  X <- c()
  t <- 1
  X[t] <- init
  while(t < 10000){
    Y <- rlnorm(1, mean = X[t])
    u <- runif(1)
    if(u <= alpha_ln(X[t],Y)){
      X[t+1] <- Y
    }else{X[t+1] <- X[t]}
    t <- t+1
  }
  return(X)
}

mh <- mh()
ts_mh <- ts(mh)
plot(time(ts_mh),mh, type = "l")
alpha_cs <- function(x, y){
  q_x_y <- dchisq(x, floor(y+1))
  q_y_x <- dchisq(y, floor(x+1))
  a <- min(1, pi(y)/pi(x)*q_x_y/q_y_x)
  return(a)
}

mh_2 <- function(init = 1){
  X <- c()
  t <- 1
  X[t] <- init
  while(t < 1000){ # only 1000 points this time, so it is more readable
    Y <- rchisq(1, floor(X[t]+1))
    u <- runif(1)
    if(u <= alpha_cs(X[t],Y)){
      X[t+1] <- Y
    }else{X[t+1] <- X[t]}
    t <- t+1
  }
```

```r
  return(X)
}

mh2 <- mh_2()
ts_mh2 <- ts(mh2)
plot(time(ts_mh2),mh2, type = "l")
library(coda)
mcmc <- data.frame(x = rep(0,10000))
for(i in 1:10){
  mcmc <- cbind(mcmc,mh_2(i))
}
mcmc <- mcmc[,-1]
names(mcmc) <- 1:10

f <- mcmc.list()
for(i in 1:10) f[[i]] <- as.mcmc(mcmc[,i])
gelman.diag(f)
load("chemical.RData")
chem<-data.frame(X,Y)
plot(X,Y)
gibbs.sampler<-function(data,k){
  #k is the values we want to obtain using the function.
  n<-dim(data)[1]
  Y<-data$Y
  values<-matrix(rep(0,k*n),nrow=k)
  mu<-rep(0,n)
  iter<-0
  while(iter<k){
    iter<-iter+1
    mu[1]<-1
    for (i in 2:(n-1)){
      u<-(1/3)*(mu[i+1]+Y[i]+mu[i-1])
      mu[i]<-rnorm(1,u,0.2)
    }
    u<-(1/2)*(Y[n]+mu[n-1])
    mu[n]<-rnorm(1,u,0.2)
    values[iter,]<-mu
  }
  return(values)
}
Z<-gibbs.sampler(chem,1000)
z<-apply(Z,2,mean)
par(mfrow=c(1,1))
plot(X,Y)
points(X,z,col="red")
plot(Z[,50],type="l")
setwd("/Users/niclaslovsjo/Library/Mobile Documents/com~apple~CloudDocs/Kurser/ML/t7")
## NA
```